Scientific Computation Notes

Taylor Series: $\sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n$

With Remainder: $f(x) = \sum_{n=0}^{m-1} \frac{f^{(n)}(a)}{n!}(x-a)^n + \frac{f^{(m)}(c)}{m!}(x-a)^n$ for some $c \in [x, a]$

$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \dots$

$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \dots$

$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \dots$

$\ln(1-x) = -x - \frac{x^2}{2} - \frac{x^3}{3} - \frac{x^4}{4} - \dots$

$\ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \dots$

$\arctan(x) = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \dots$

Memory

Adds And Multiplies

$P(x) = x^7(a_7 + x^5(a_{12} + x^5(a_{17} + x^5(a_{22} + x^5 \cdot a_2 7))))$

Binary Exponentiation

Repeating Representation: $\frac{a}{b}$ in base c need $\frac{a}{b} = \sum d_i c^i + c^j \cdot \frac{d_h}{c^k - 1}$ with $b | c^{h'}(c^k - 1)$ so relevant $\phi(p) = p - 1$ from prime factorisations lead to k and then fractional representation conversion.

IEEE Double Precision Representation For Floating Point Number: 1 Sign Bit 11 Exponent Bits 52 Mantissa Bits, $\epsilon_{\text{machine}} = 2^{-52}$

Chopping, Rounding [Towards 0, Towards Even], Nearest: if the 53rd bit is 0 then round down truncate, if $1.\dots100\dots0\dots = 1.\dots1\overline{0}$ then round down truncate, else round up add 1 to 52nd bit and reoperate. Can express 9.4 as $+1.0010110011\dots101 \times 2^3$.

```
Round Down
1.00...000     +
 .       110100 =
1.00...011
Round Down
1.11...111     +
 .       001000 =
1.11...111
Round Up [?] In Register
1.11...111     +
 .       001100 =
1.00...000 x 2^1
```

Relative Error $\frac{|\text{float}(x) - x|}{|x|} \leq \frac{1}{2}\epsilon_{\text{machine}}$ For $x \neq 0$

Bisection Method i.e. Binary Searching: if one has an interval $[a, b]$ which contains a root e.g. $f(a)f(b) < 0$ then iteratedly replace the proper endpoint with $\frac{a+b}{2}$ thereby converging linearly upon the root r with factor $S = \frac{1}{2}$.

Fixed Point: $f(x) = x$

Root Multiplicity: Intuitive $(x-r)^{\text{multiplicity}}$ term in finite polynomial functional representation, degree of derivative where $f^{(\text{multiplicity})}(r) \neq 0$, lowest degree term in Taylor Series around r.

Fixed Point Iteration:
Want Fixed Point Or Root Of $f(x)$ e.g.

Fixed Point Of Naive $g(x) = f(x) + x$
Choose Rearrangement Carefully
$x_0 =$ initial guess
$x_{i+1} = g(x_i)$
Linear Convergence
Meaning Errors Satisfy
$\lim_{i\to\infty} \frac{e_{i+1}}{e_i} = S = |g'(r)| < 1$

Root Finding Problem
Forward Error: $|r - x_a|$
Backward Error: $|f(x_a)|$

Sensitive, Small Errors In Input Lead To Large Errors In Output, Error Magnification Factor, Condition Number

Sensitivity Formula For Roots: if $\epsilon << f'(r)$, r is a root of $f(x)$ and $r + \Delta r$ is a root of $f(x) + \epsilon g(x)$ then $\Delta r \approx -\frac{\epsilon g(r)}{f'(r)}$.

error magnification factor = $\frac{\text{relative forward error}}{\text{relative backward error}}$

Newton's Method:
$x_0 =$ initial guess
$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$
Quadratically Convergent If Root Has Multiplicity 1 i.e. $f'(r) = 0$.
Meaning Errors Satisfy
$\lim_{i\to\infty} \frac{e_{i+1}}{e_i^2} = M, M = \frac{f''(r)}{2f'(r)}$ Otherwise Linear
Multiplicity m Root r:
$\lim_{i\to\infty} \frac{e_{i+1}}{e_i} = S = \frac{m-1}{m}$
Modified
$x_{i+1} = x_i - m \cdot \frac{f(x_i)}{f'(x_i)}$
Quadratically Convergent

Failure

Can blowup diverge to infinity $f(x) = x^{\frac{1}{3}}$ or fail due to divide by 0 if $f'(x_i) = 0$.

Secant Method:
$x_0, x_1 =$ initial guesses
$x_{i+1} = x_i - \frac{f(x_i)(x_i - x_{i-1})}{f(x_i) - f(x_{i-1})}$

Method Of False Position:

```
for i=1,2,3,...
    c=(bf(a)-af(b))/(f(a)-f(b))
    if f(c)=0,stop,end
    if f(a)f(c)<0
        b=c
    else
        a=c
    end
end
```

Muller's Method: use 3 previous points, parabola, nearest root to previous point is next point. Complex arithmetic software complex roots. Faster convergence than Secant Method.

Inverse Quadratic Interpolation: use 3 previous points, parabola function in y, Lagrange Interpolation. Faster convergence than Second Method.

Brent's Method: hybrid method, Matlab fzero command.

Gaussian Elimination: $O(n^3)$

Reduced Row Echelon Form: can augment with b_i to resolve $Ax = b_i$.

Lower Upper Factorisation: add copies of row 1 to rows $2, 3, \ldots$ to zero column

1 below the diagonal in U and store those coefficients in column 1 of L, iterate. For the back substitution, solve $Lx' = b_i$ and then $Ux = x'$.

Compute Estimate: $\approx \frac{2}{3} \cdot n^3$ and $2n^2$ operations for Lower Upper Factorisation and each instance of computing x such that $Ax = b_i$ for a total of $\frac{2}{3} \cdot n^2 + 2n^2k$.